DEOS Symposium 2012

Open Systems Dependability

A New Approach to Attain Dependability of Ever-Changing Huge and Complex Software Systems

March 7, 2012

Mario Tokoro

Research Supervisor, JST/CREST Dependable Embedded OS for Practical Use

Chairman and CEO, Sony Computer Science Laboratories, Inc.

科学技術振興機構
Japan Science and Technology Agency

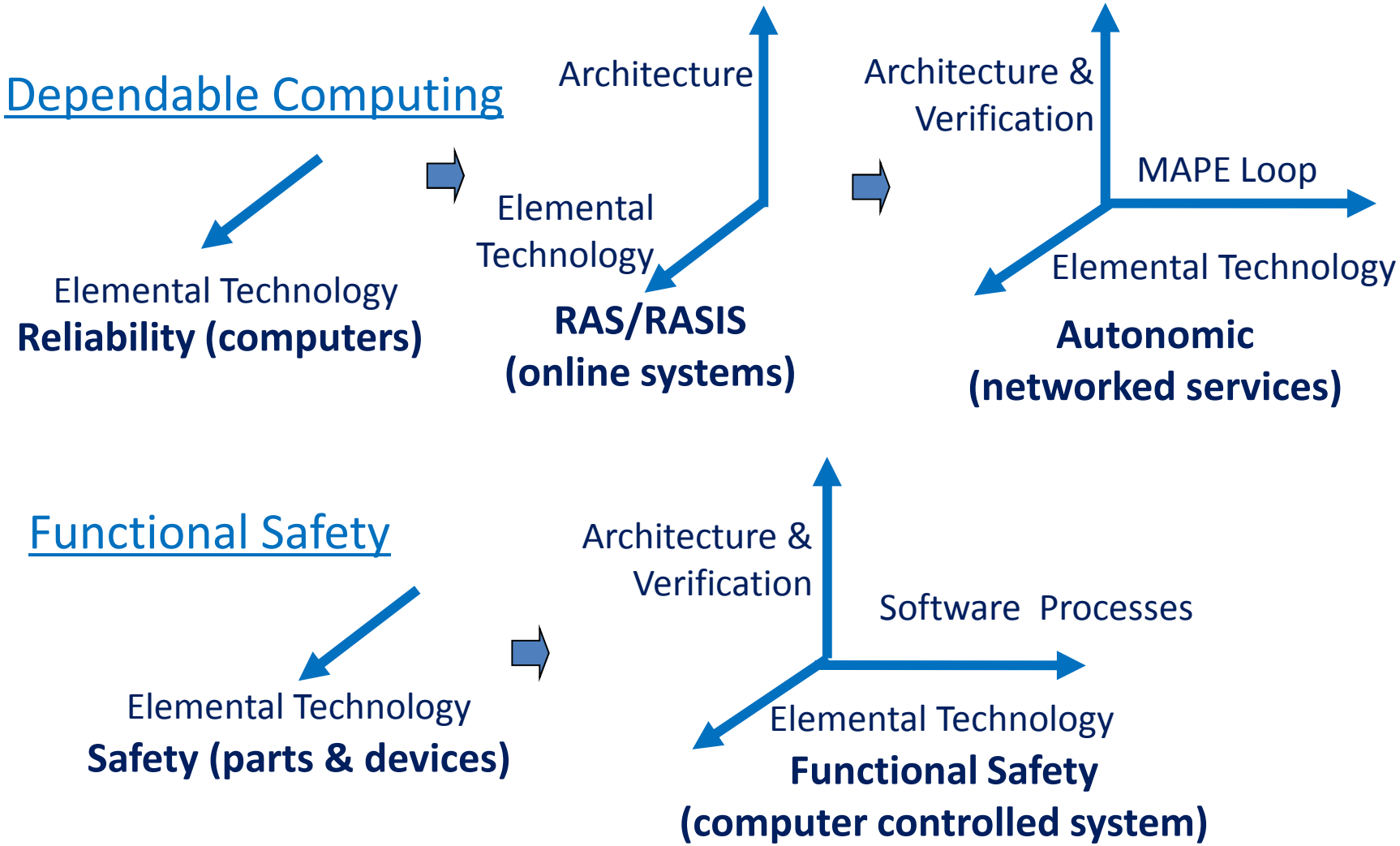
Recent Information Systems

- Huge and Complex, using Acquired Components and External Services
- Used for a long period of time
- Modification are continuously performed

- Gap grows between
  - Intention and Specification
  - Specification and Implementation
  - Implementation and Operation
  - Operation and Operation Manuals
  - With non-determinacy property (ambiguity of meaning)

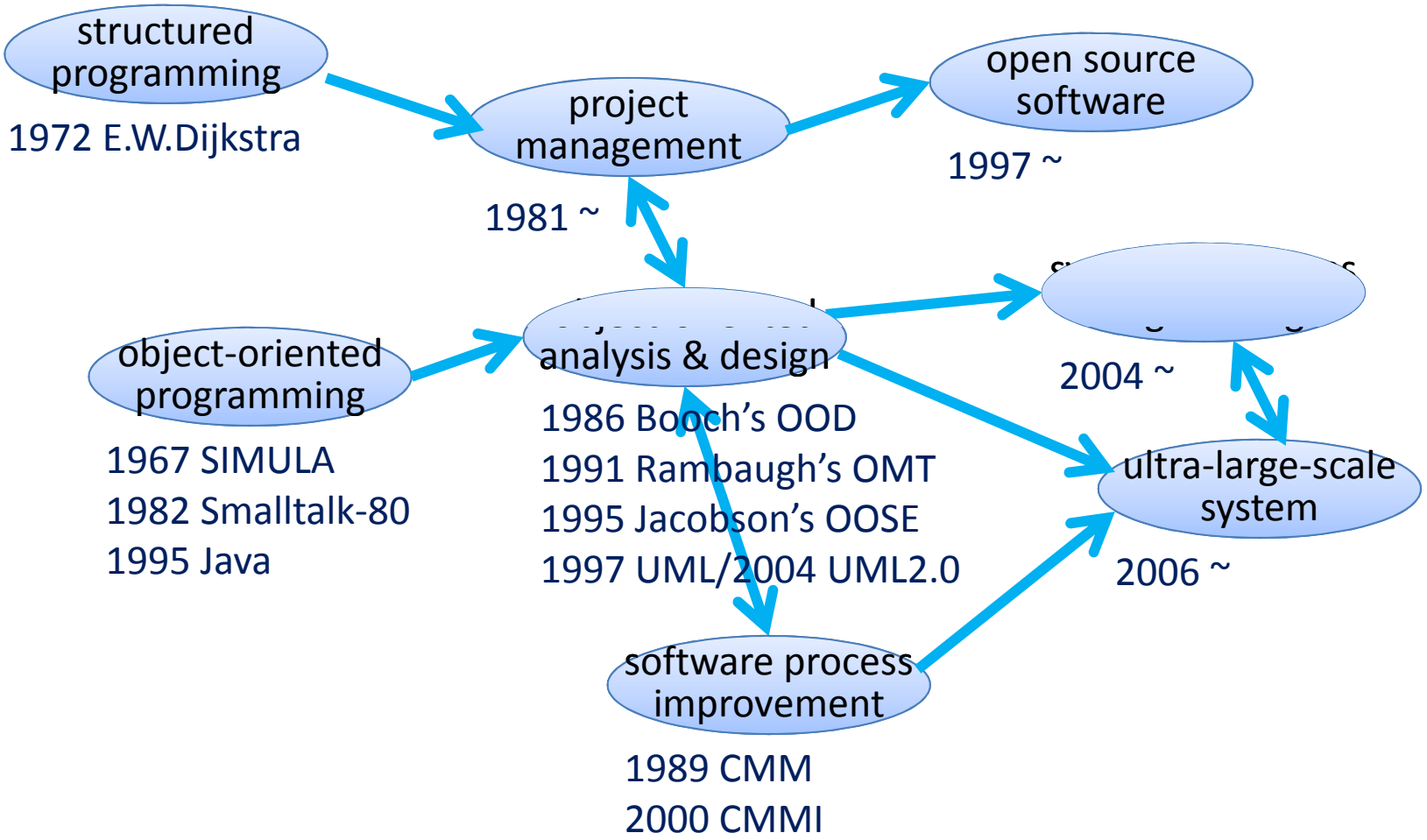
Dependability

A Brief Historical Review



Software Engineering

A Brief Historical Review



What is Dependability?

Definition by IFIP WG10.4 (2004)

Standards and Guides

- Standards
  - IEC 61508: Functional Safety
  - IEC 60300-1: Dependability Management
  - IEC 60300-2: Dependability Program Elements and Tasks
  - ISO/IEC 1207: Software Life Cycle Processes
  - ISO/IEC 15288: System Life Cycle Processes
  - ISO/FDIS 26262 Road Vehicles Functional Safety
  - etc.
- Guides
  - CMMI: Capability Maturity Model Integration
  - DO-178B: Software Considerations in Airborne Systems and Equipment Certification
  - MISRA-C: Guidelines for the Use of the C Language in Vehicle Based Software
  - IEC 61713: Software Dependability through the Software Life-Cycle Processes – Application Guide
  - IEC 62347: Guidance on System Dependability Specifications
  - etc.

Industry Initiatives

- AUTOSAR for Automotive
- HIDENETS for Automotive
- ARINC 653 for Aerospace and Defence
- Solaris 10, Linux RAS, and Autonomic Computing for Enterprise Systems
- Carrier Grade Linux and Service Availability Forum for Communication/Network Operators
- The Open Group / TOGAF etc.

Recent Incidences in Japan

- August 10, 2010, MIXI (largest SNS in Japan) stopped for two days
- February 24, 2009, Google Apps Gmail service stopped
- September 14, 2008, Airports check-in terminal could not continue services
- July 22, 2008, Trading systems for derivatives down
- October 12, 2007, IC ticket gates service stopped
- September 19, 2006, October 23, 2006, May 16, 2007, and May 23, 2007, Internet Service stopped
- May 27, 2007, ticketing and Check-in systems stopped

Existing Methods for Dependability

- Consideration for Elemental Technology
- Consideration for Architecture
- Consideration for Development Process

- Less Consideration for Modification after Initial Development
- Less Consideration for Operation (Relation among Implementation/ Operation/ Operation Manuals)

- Most of Recent Incidences occur due to Less Consideration of Changes

Demands

- Strong demands for the dependability of huge and complex software systems
  - which include black box software such as legacy codes and off-the-shelf components
  - which are connected to networks that may cause security and integrity problems
- Increased demands for coping with environmental and requirement changes in operation
  - functions, user interfaces, performance, etc
  - networks and services on networks
- Necessity of Continuous Operations
- Increased accountability to service/system providers

Properties of Modern Systems

- We cannot assume that a system is composed of *complete* components
  - No, because we use legacy/off-the-shelf components developed outside
- We cannot assume that we can know the *boundary* and the *behaviors* of the whole system
  - No, because we use network services
  - No, because the system is ever-changing due the change of service objectives, users' needs, and environmental (servers/networks) changes

Open Systems View

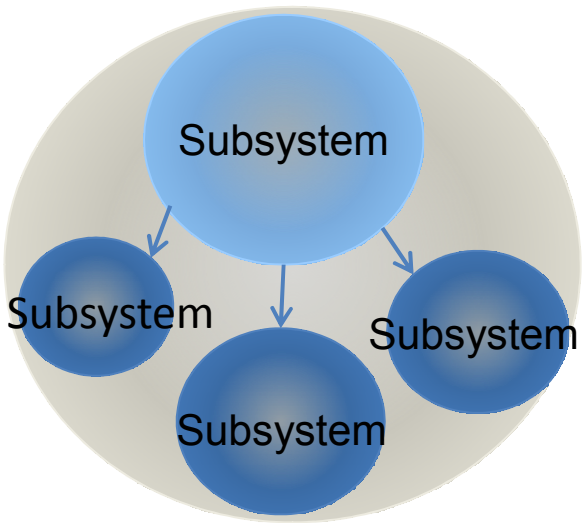
- Thus, we need to view a huge, complex, and ever-changing system as an Open System
  - Open System ≠ Open Source Software
  - Open System ≠ Open Architecture
- The notion of Open System is a higher one to conceptualize the systems in which the boundary, functions, structures, and interfaces are ever-changing

Most Important Issue

- Change Management
  - For the change of objectives
  - Users requirement change
  - Hardware/Network/Service change

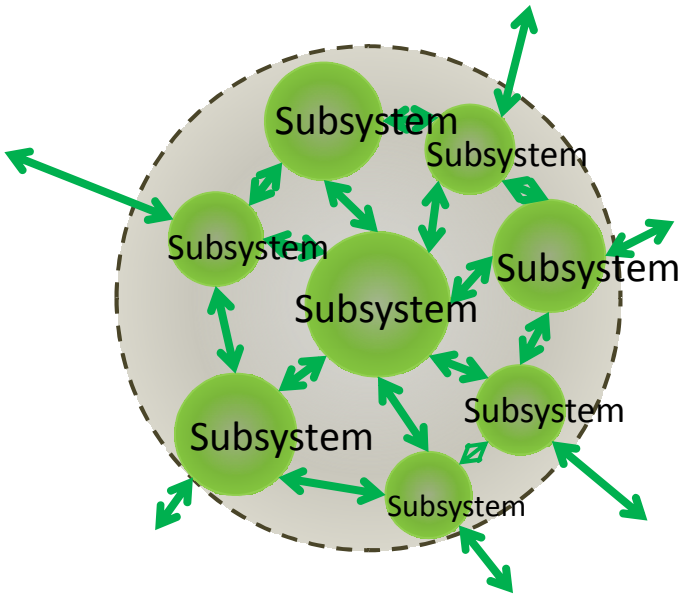
Closed Systems vs. Open Systems

Closed Systems



- The boundary of the system is definable.
- Interaction with the outer world is limited, and the system functions are fixed.
- The subsystems or components of the system are fixed and their relationship does not change over time.

Open Systems



- The boundary of the system changes over time.
- Interaction with the outer world and the system functions change over time.
- The subsystems or components of the system and their relationship change over time.

Can We Consider a Huge, Complex, and Ever-Changing Software System as a Closed System?

Yes, if we can assume:

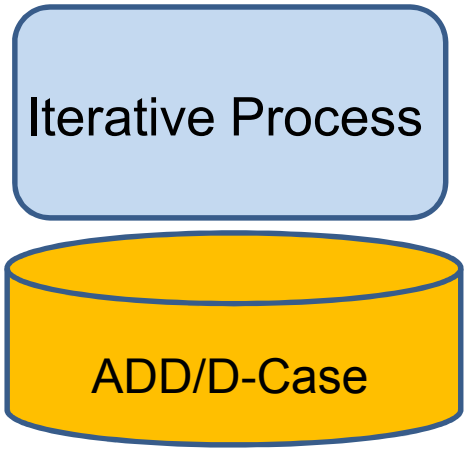
- a system does not change for a certain period of time and
- the lifecycle of the system can be the accumulation of these periods of time.

However, if we need the system continues to give services while being modified, and possibly even in the case of incidence, it is extremely difficult to separate the phase that

- the system is in normal operation
- the system is being fixed (for incidence) and
- the system is being modified (for requirement and environmental changes).

Can We Achieve Dependability to Open Systems

- With Elemental Technologies and Architecture?
  - No. But by an Iterative Process to cope with an ever-changing system
  - With the records of all the stakeholders' agreements and evidences in Agreement Description Database / D-Case



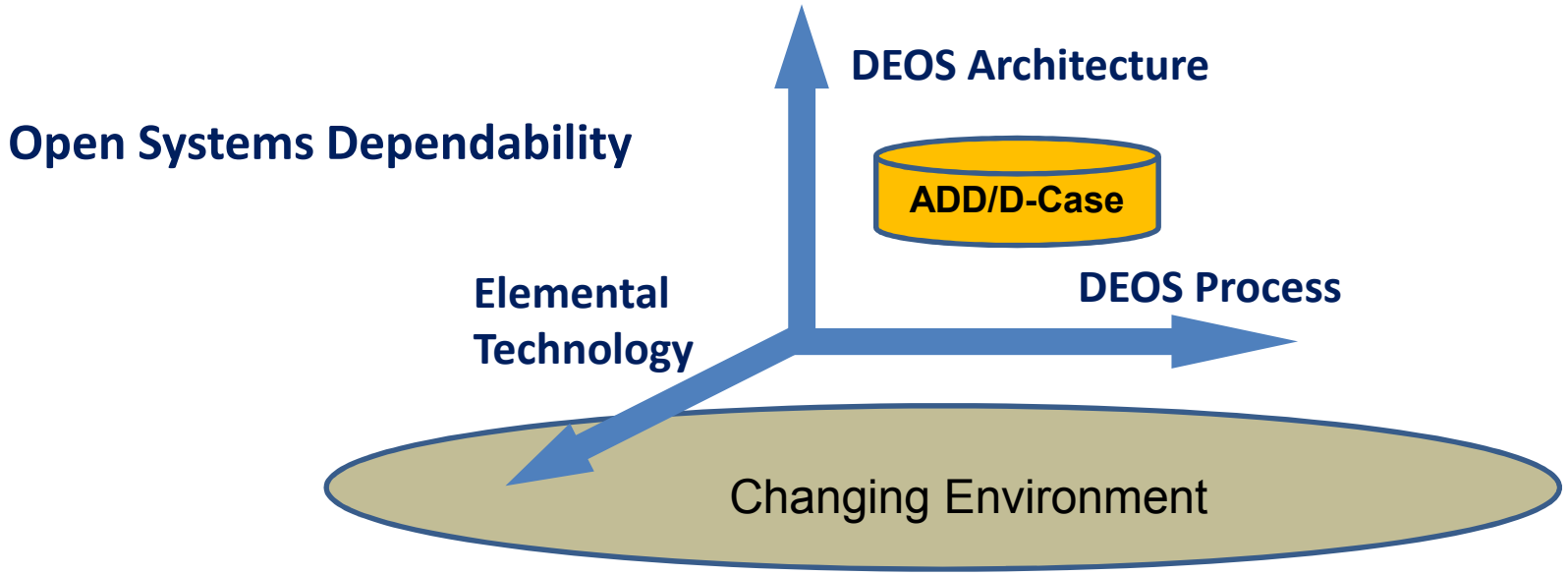
Open Systems Dependability: Definition

- Functions, structures, and boundaries of a huge and complex software system change over time. Hence, incompleteness and uncertainty are inherent to such a system, which may result in failures in the future.
- Open Systems Dependability is a property of a system such that it has the ability
  1. to continuously prevent those problem factors from causing failure,
  2. to take appropriate and quick action when failures occur to minimize damage,
  3. to safely and continuously provide the services expected by users as much as possible, and
  4. to maintain accountability for the system operations and processes.

The DEOS Process

The DEOS Process is a process to achieve Open Systems Dependability by

- Iterative process (DEOS Process)
- with ADD/D-Case
- which are supported by architecture (DEOS Architecture)
- which is supported by elemental technologies

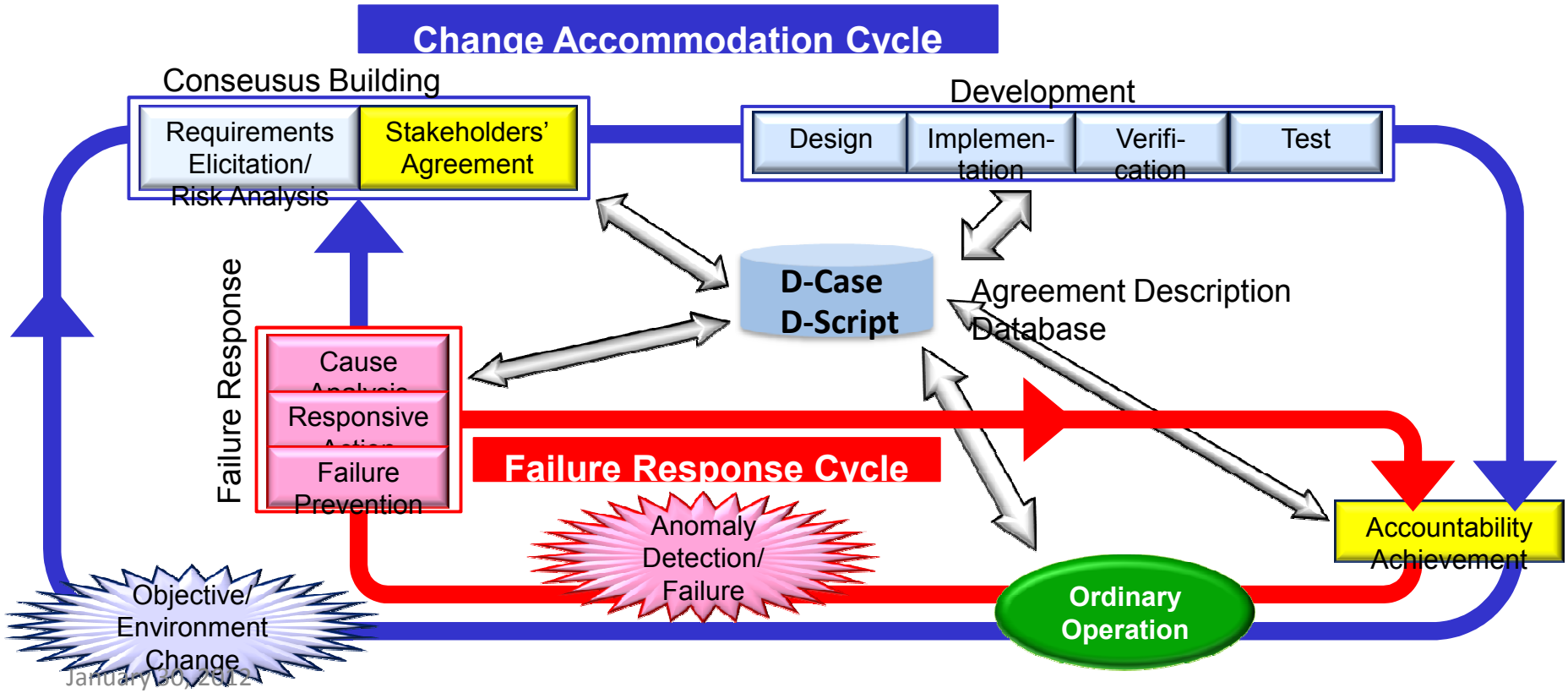


Short Video

The DEOS Process

A process to achieve Open Systems Dependability

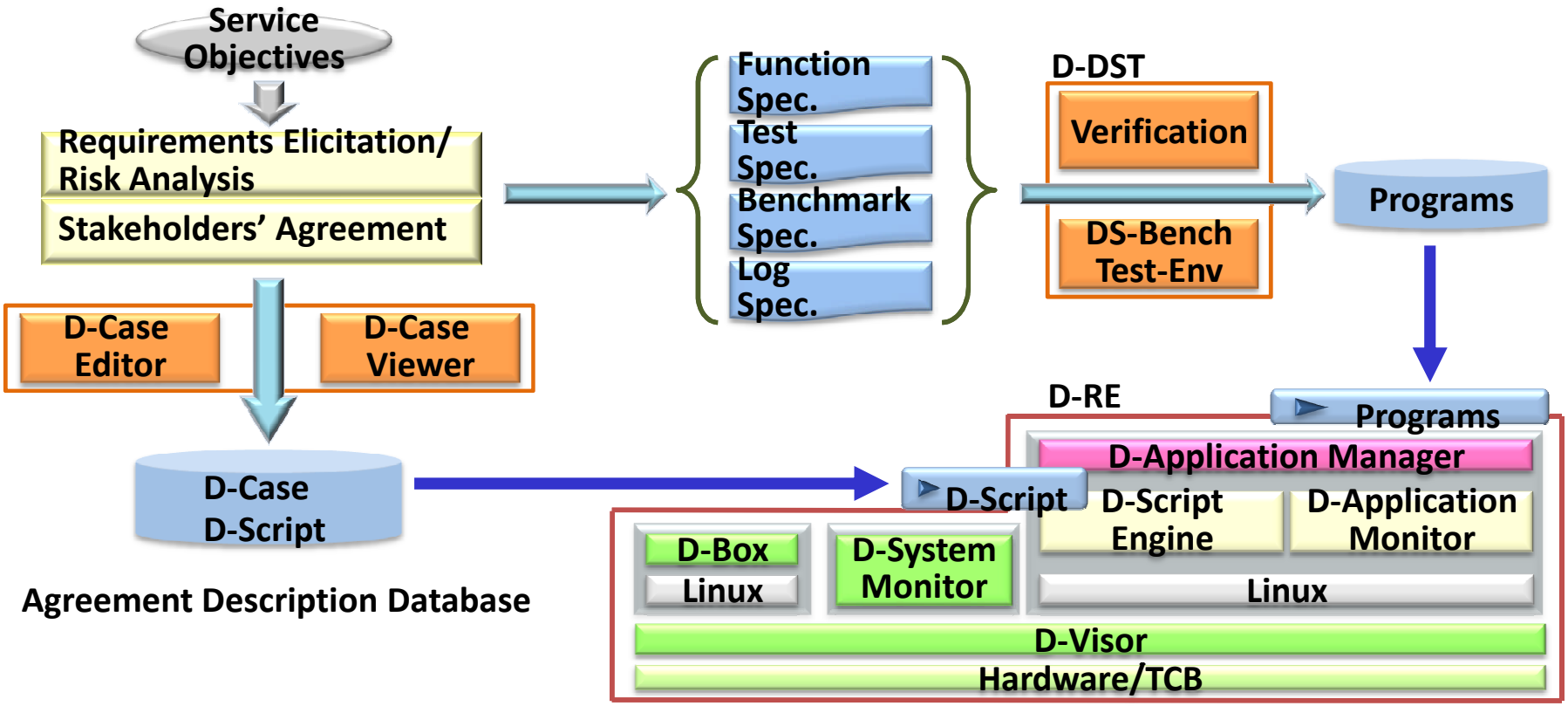
- Iterative process
  - Change Accommodation Cycle to accommodate requirement changes in service objectives and environments
  - Failure Response Cycle to respond quickly and properly to failures
- A process of processes that are organically connected



DEOS Architecture

A DEOS architecture supports the execution of the DEOS process

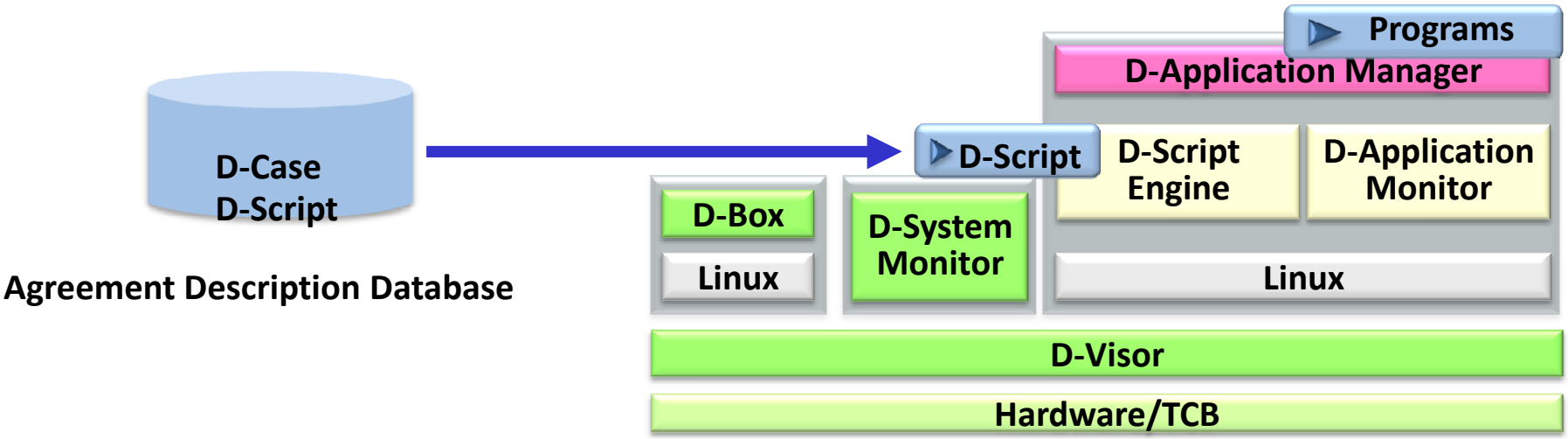
- Agreement Description Database and tools to support requirements management
- Tools to develop dependable software (D-DST)
- Execution Environment to execute programs, to monitor and record the states of programs, and to respond to failures (D-RE)
- A DEOS architecture is application region specific



DEOS Runtime Environment (D-RE)

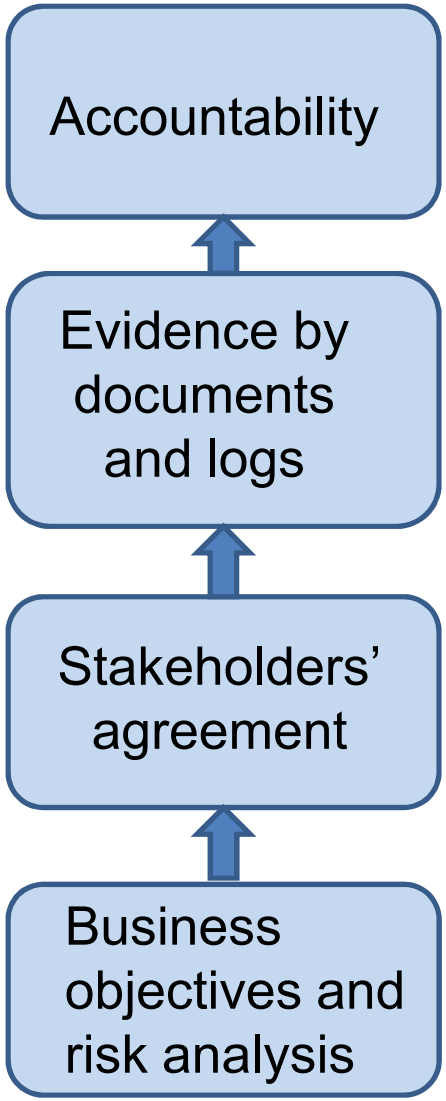
D-RE is application/implementation specific. A D-RE can consist of

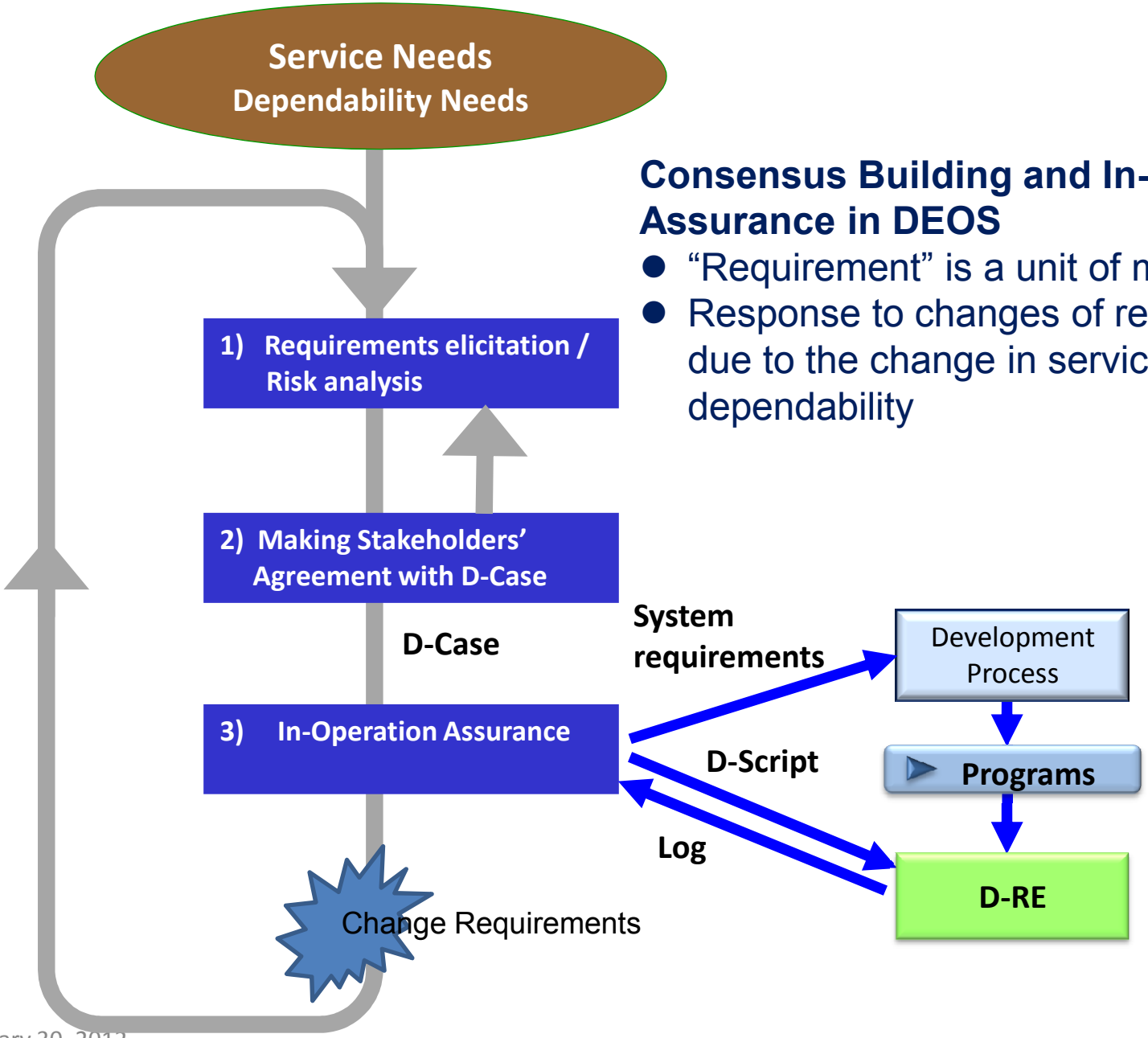
- D-Application Manager provides application containers
- D-Application Monitor monitors applications and gathers logs
- D-System Monitor monitors the system and gathers logs
- D-Script contains scenarios that designate D-RE when and what parameters to log and how to respond to failures
- D-Script Engine executes D-Script safely
- D-Box preserves logs and functions as a flight recorder
- D-Visor abstracts hardware and provides system containers



Accountability Achievement in the DEOS Process/Architecture

- In order to provide *accountability,* we need to show *evidence* by
  - process documents e.g. design, implementation, verification, test, failure prevention, responsive action, cause analysis, and daily check and maintenance, and
  - logs that record the behavior of the system given by D-RE
- Which are collected based on *stakeholders' agreement* through the argumentation and documentation tools called D-Case
- Which are supported by business objectives and risk analysis





Consensus Building and In-Operation Assurance in DEOS

- "Requirement" is a unit of management
- Response to changes of requirements due to the change in service and dependability

Making Stakeholders' Agreement

Who are Stakeholders?

- Service/Product Users
- Service/Product Providers
- System Providers
  - Software Developers/Integrators
  - Operators/Maintainers
  - Hardware Providers
- Authority for Service/Product

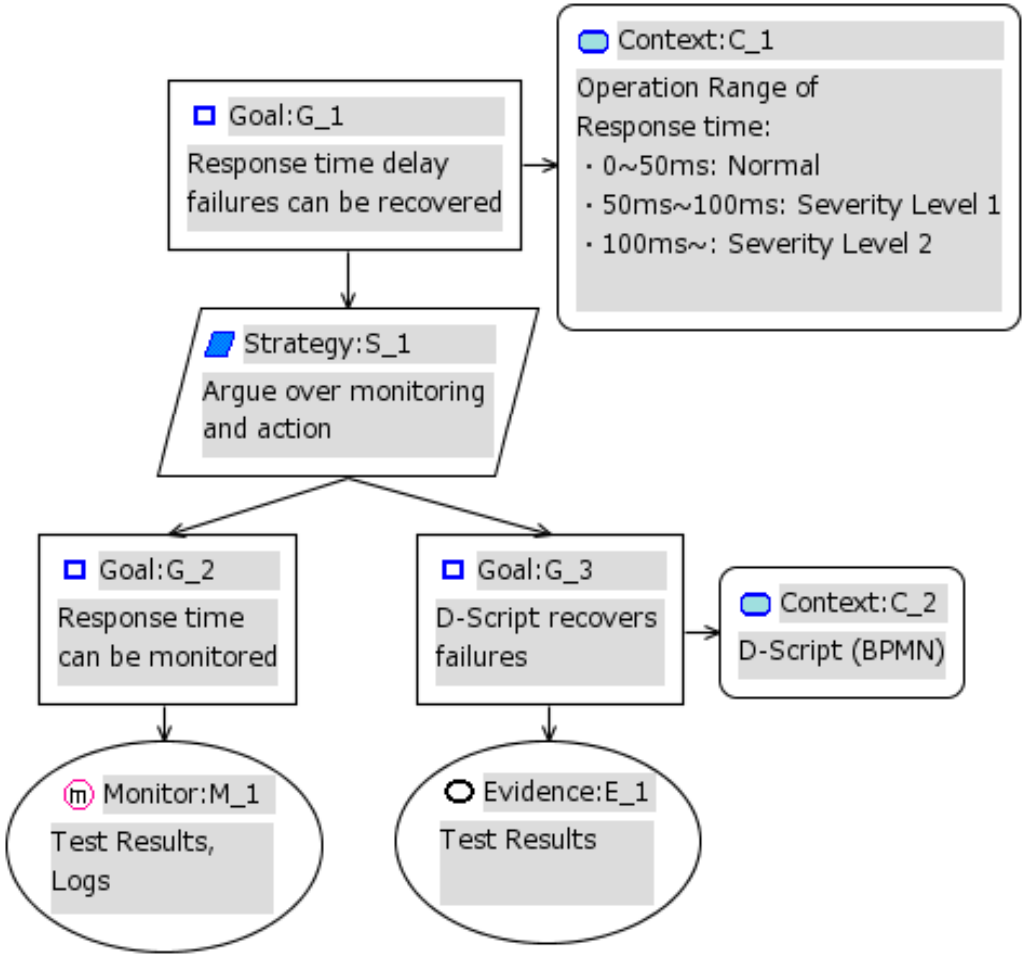
Stakeholders' agreement is performed using D-Case. D-Case is developed based on GSN (Goal Structuring Notation), and has following characteristics:

- Structured notation for consensus building through argumentation and evidence
- Management support for stakeholders' agreement
- Designation of monitoring in operation to use as evidence
- Consistency checking for the agreements

D-Case Example

An example of argumentation for response delay in a server

- Context node C_1 designates the acceptable deviation of response time to be from 0 to 50msec and the severity level to be 1 and 2.
- Monitor node M_1 designates server's response time is monitored.
- Evidence node E_1 assures by test results that when the response time exceeds the acceptable deviation, D-Script takes action to fix the situation.



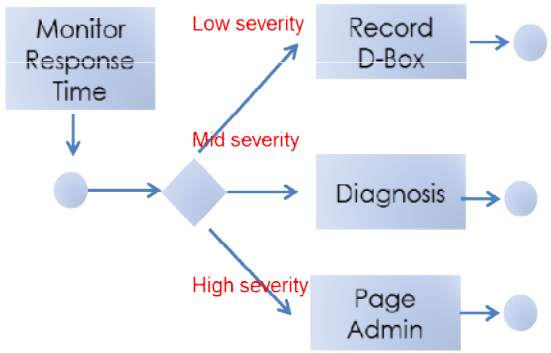
In Operation Assurance through D-Script

D-Script designates logging and checking dynamically based on D-Case description and with the help of D-RE (D-System Monitor, D-Application Monitor, and D-Box)

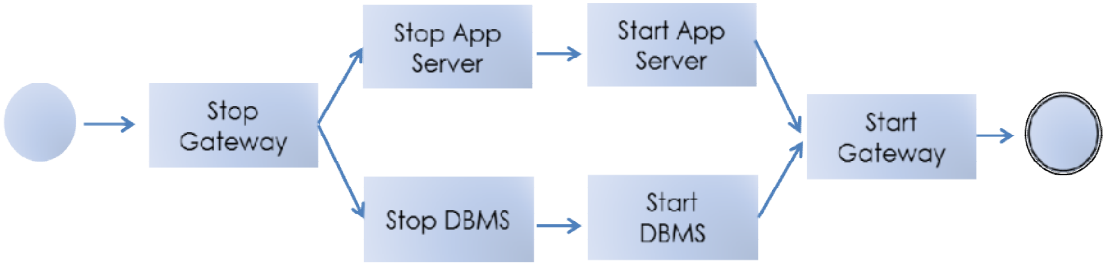
- D-Script: a set of D-Script Scenarios
- D-Script Scenario: a procedure consisting of D-Tasks and D-Controls to be executed by D-Script Engine
- D-Task: a primitive action
- D-Control: designation of sequential, branching, and parallel execution

D-Task

D-Control



D-Script Scenario



Evaluation of Conformance to DEOS Process

Evaluation in each life stage in DEOS Process State and Cycles

DEOS process State and Cycles	Life Stage: Planning	Life Stage: Implementation	Life Stage: Execution
Ordinary Operation	Evaluation criteria for in-operation plan	Evaluation criteria for in-operation implementation	Evaluation criteria for in-operation result
Failure Response Cycle	Evaluation criteria for failure response plan	Evaluation criteria for failure response implementation	Evaluation criteria for failure response result
Change Accommodation Cycle	Evaluation criteria for change accommodation plan	Evaluation criteria for change accommodation implementation	Evaluation criteria for change accommodation result

The New Notion of DEOS

- DEOS is no just Dependable Embedded OS
- It is the Process to achieve Open Systems Dependability together with its Architectures and Elemental Technologies

- Now, DEOS stands for

  Dependability Engineering for Open Systems

Standardization

Purpose

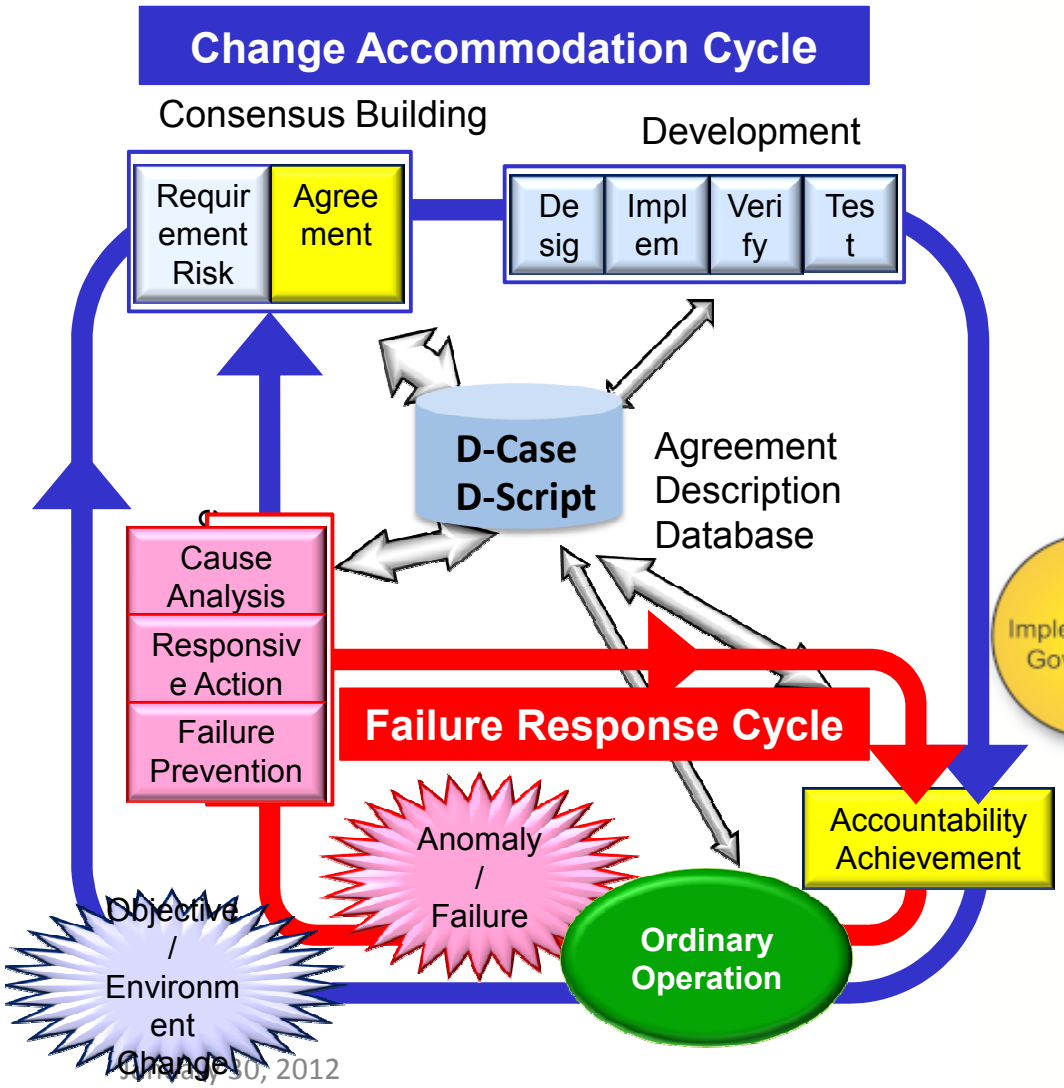
- **Sharing the concepts of Open Systems Dependability**
- **Provide guidelines for social infrastructures**
- **Achieving** common use of tools

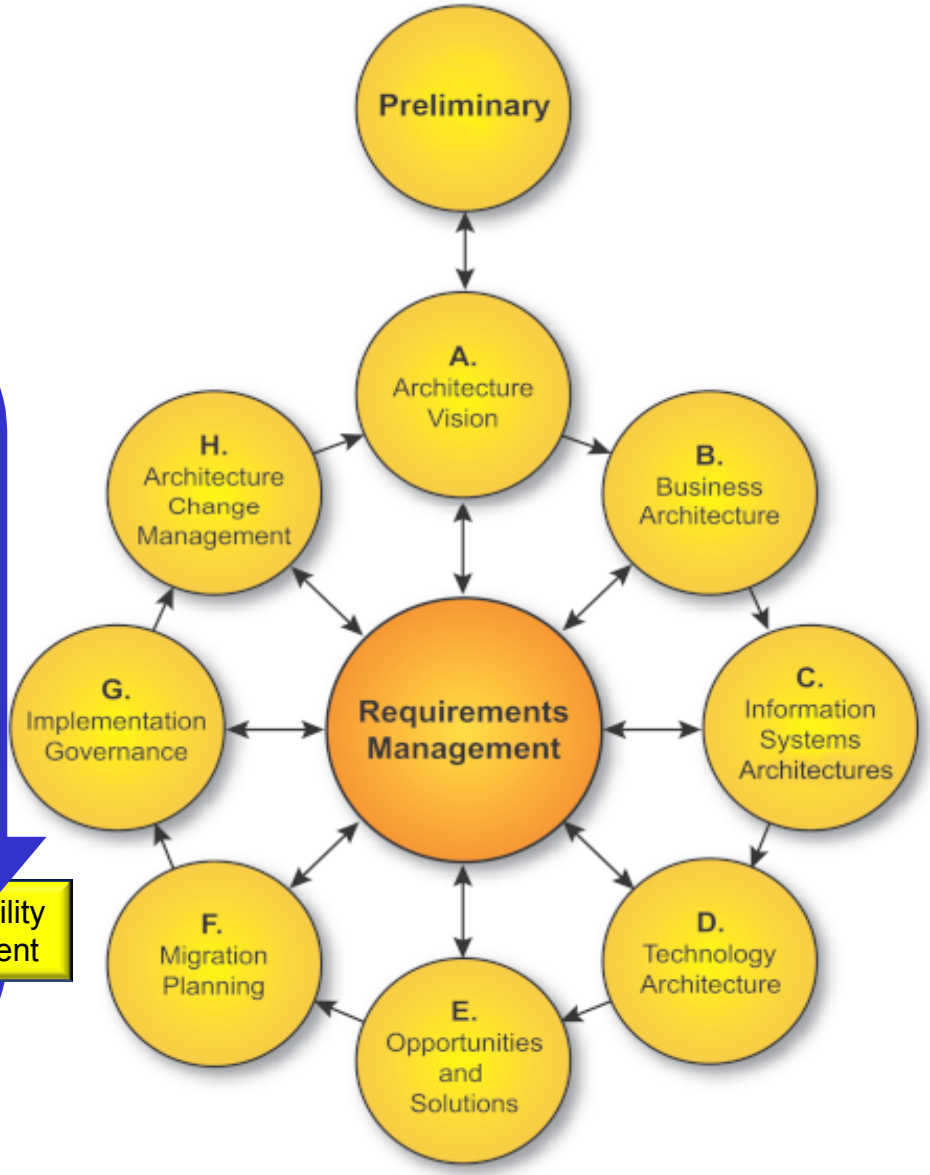
Plans

- **Standard for the concept of Open Systems Dependability**
  - **IEC60300, IEC TC56 NWIP**
- **Standards for methodology of consensus building, achieving accountability and process**
  - **ISO/IEC15026 System and Software Assurance**
  - **IEC TC56**

- **At the same time, we would like to discuss in The Open Group, because DEOS Process is compatible and/or supplementary with TOGAF**

DEOS and TOGAF





Summary

- A huge, complex, and ever-changing system inherently has incident factors due to *incompleteness* and *uncertainty*
- We proposed a new approach called *Open Systems Dependability*
- Open Systems Dependability is achieved by the *DEOS Process*, with ADD/D-Case, supported by the *architecture* and *elemental technologies*

Plan and Schedule

- JST fund for two more years (until March 2014)
- We are benchmarking the effectiveness of the DEOS process by describing D-Case with a few companies
- A book on OSD/DEOS will be published in 2012

Contributors

Shigeru Matsubara	Dependable Embedded OS R&D Center, JST
Tomohiro Miyahira	Dependable Embedded OS R&D Center, JST
Kiyoshi Ono	Dependable Embedded OS R&D Center, JST
Hiroki Takamura	Dependable Embedded OS R&D Center, JST
Makoto Yashiro	Dependable Embedded OS R&D Center, JST
Kenji Kono	Keio University
Jin Nakazawa	Keio University
Hideyuki Tokuda	Keio University
Hiroshi Yamada	Keio University
Shuichiro Yamamoto	Nagoya University
Yoichi Ishiwata	National Institute of Advanced Industrial Science and Technology
Satoshi Kagami	National Institute of Advanced Industrial Science and Technology
Yoshiki Kinoshita	National Institute of Advanced Industrial Science and Technology
Toshinori Takai	National Institute of Advanced Industrial Science and Technology
Makoto Takeyama	National Institute of Advanced Industrial Science and Technology
Kenji Taguchi	National Institute of Advanced Industrial Science and Technology
Mario Tokoro	Sony Computer Science Laboratories, Inc.
Hajime Fujita	University of Tokyo
Yutaka Ishikawa	University of Tokyo
Toshiyuki Maeda	University of Tokyo
Yutaka Matsuno	University of Tokyo
Yasuhiko Yokote	University of Tokyo
Taisuke Boku	University of Tsukuba
Toshihiro Hanawa	University of Tsukuba
Shuichi Oikawa	University of Tsukuba
Mitsuhisa Sato	University of Tsukuba
Tatsuo Nakajima	Waseda University
Kimio Kuramitsu	Yokohama National University
Midori Sugaya	Yokohama National University